# A METHOD FOR TRANSFORMING DATA FORMATS BETWEEN DIFFERENT DATABASE SYSTEMS, AN APPARATUS FOR EXECUTING THE METHOD AND THE PROGRAM OF THE METHOD

## BACKGROUND OF THE INVENTION

### Field of the Invention

The present invention relates to a method for transforming data formats between different database management systems and an apparatus for executing the method, and more particularly to a method for transforming data formats between different database management systems, which needs no data transfer between a host computer and a disk storage device to reduce the system load when transforming a large scale database, and an apparatus for achieving the method.

### Description of the Related Art

In the decade of 1990, data intensive applications have been emerged, such as data mining, data warehouse, and decision support system, which may process large amount of data. In such a situation, the amount of data doubles year by year, solutions adapted for efficiently managing data have been demanded. SAN (Storage Area Network) is one of solutions proposed in the second semester of 1998.

SAN is a network dedicated for data transfer, composed of storages and computers that access the storages. For example, data backup was done by using a LAN connecting other computers.

When using SAN, the network dedicated for data transfer, the load traffic on the LAN can be reduced. The reduction of load on the LAN is one of major purposes of SAN. SAN may also be characterized by easy data sharing. This is because computers connected to SAN have physically access to any magnetic disk drives connected thereto.

However, when two computers can physically access to one same magnetic disk drive, it does not necessarily mean that the data can be shared at the application level. Data that is managed by a database management system (DBMS herein below) or a file system on one of those computers may be accessed by another computer, however another computer may have no means to interpret it. For this reason a variety of converter softwares have been developed for achieving data sharing between a file system and a DBMS or between different DBMS.

Data mining is often discussed as a method of effective exploitation of huge amount of data and tools for data mining are actively developed. In general, data mining tools may use data (for example, consumers' data) stored by OLTP (Online Transaction Program). An OLTP usually runs on a mainframe, and uses a DBMS for managing data. A data-mining tool, on the other hand, runs on an open system such as Unix or Windows NT , and analyses data after storing data into a DBMS. Here lies the necessity of data transfer from a mainframe to an open system and data conversion between different DBMS.

As known techniques of data conversion method between different DBMS there are discloses such as United States Patent No. 6016501 and 6035307.

An EDM system (Enterprise Data Movement) system, cited in the above patent application No. 6016501 extracts data from the source DBMS to transform data format to that of targeted DBMS and feed the transformed data to the target DBMS. In general, data of the source DBMS and that of target DBMS are stored in a disk storage device, and the EDM system runs on a server. The data of source DBMS will be extracted to the server from the disk storage device through a SCSI channel, transformed to the data format specified by the target DBMS on the server, and loaded to the data field of the target DBMS through the SCSI channel.

Fig. 11 shows schematically this method.

Fig. 11 shows a schematic diagram illustrating a data conversion method in accordance with the Prior Art.

In the data conversion as shown in Fig. 11, data in a DB 1 format, stored in a disk 200A of a disk storage device 120 will be loaded into a Unix host computer 100B, transformed to data in a DB 2 format by the data extraction/ conversion/ loading program to write into the disk 200B.

The data transfer between server and disk storage occurs twice here (once for reading out source data, and once for writing down the transformed data).

For the purpose of performing data mining, the amount of data transferred from the mainframe to the Unix host can easily reach to a few Tbytes (terabytes). This amount can be otherwise described a 10-hours course using a fibre channel of 100 Mbytes per second. The load to the entire system will be reached to an extreme.

There may be cases in which instead of one-step operation of the extraction/ conversion/ loading from the source DBMS data format to the target DBMS data format, the operation may be performed in three separated steps of extraction, conversion, and loading. In Fig. 11 of the aforementioned United States Patent No. 6035307, an example of the Prior Art is cited, which perform database format conversion via a few intermediate-working formats.

A database format conversion using some intermediate file formats will be described here by referring to Fig. 12.

Fig. 12 is a schematic diagram illustrating an exemplary data conversion in accordance with the Prior Art.

In a mainframe 100A, there is an extractor program, which transforms DB 1 format data in a disk 200A to the format 1 data on the disk 200B. On a Unix host 100B a transformer program and loader program are installed, the transformer program transforms the format 1 data on the disk 200B to the format 2 data on a disk 200C, while the loader program transforms the format 2 data on the disk 200D to the DB format 2 data on the

disk 200D.

When transforming data of a database, if the intermediate data formats are used, the transformed intermediate data will also be written to the disk storage device. As a result the number of data transfer between the serve and disk storage will increase to 6 in this case, indicating the increase of data transfer time 6 times.

In the Prior Art as have been described above, data transformation is done on the host. This causes a problem that the data transformation will put some extreme load for the system. The larger the size of database is, the severer the problem becomes.

On the other hand, if the data transformation can be performed within a disk storage device, the data transfer between the server and the disk storage will be omitted.

The present invention has been made in view of the above circumstances and has an object to overcome the above problems and to provide a data transforming method of database management system, which may transform data within the disk storage device when transforming database data formats so as to reduce the system load.

Another object of the present invention is to provide a data transforming method of database management system, which may be easily developed by the program developer of the database management system and the program developer of the disk storage

device.

## SUMMARY OF THE INVENTION

In order to solve the problem cited above, the data transforming method of database management system in accordance with the present invention launches a data transformation program within a disk storage device in response to a request received from a host. The request to the data transformation program on the disk storage device may be issued by a skeleton program running on the host. The application program that can normally issue a request to the data transformation program will issue a request to the skeleton program, which in turn will pass the request to the data transformation program on the disk storage device, on the behalf of the application. By setting the interface of the skeleton program to that of the data transformation program, the execution of data transformation program may be moved to the disk storage device, without the need of changing other programs.

On the disk storage device a communication program for the skeleton program and the data transformation program to communicate each other and an I/O (input/output) program for the data transformation program to read and write data on the disk storage device are to be installed.

In general, data transforming program developers (such as DBMS vendors) between different database management systems

are quite often not the same as the program developers of the disk storage device. The program residing on a host can be developed, by the interface of communication programs and I/O programs provided by the program developers of disk storage device and by the use of such interface by the data transformation program developers.

## BRIEF DESCRIPTION OF SEVERAL VIEWS OF DRAWINGS

Fig. 1 is a schematic block diagram of network architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention;

Fig. 2 is a schematic block diagram of hardware architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention;

Fig. 3 is a schematic block diagram of software architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention;

Fig. 4 is a schematic diagram of transmission of request and reply to a program that achieves a method for transforming data formats between database management systems in accordance with the present invention;

Fig. 5 is a flow chart illustrating the steps of a method

for transforming data formats between database management systems in accordance with the present invention;

Fig. 6 is a schematic diagram of formats when a request 630 is achieved by packets of TCP/IP protocol;

Fig. 7 is a schematic diagram of a format when the request 630 is achieved by SCSI commands;

Fig. 8 is a schematic diagram of format of an I/O request 680;

Fig. 9 is a schematic block diagram of hardware and software architecture for the data transformation corresponding to Fig. 11;

Fig. 10 is a schematic block diagram of hardware and software architecture for the data transformation corresponding to Fig. 12;

Fig. 11 is a schematic diagram of an exemplary data transformation in accordance with the Prior Art; and

Fig. 12 is a schematic diagram of an exemplary data transformation in accordance with the Prior Art (using intermediate file formats).

## DETAILED DESCRIPTION OF THE INVENTION

A detailed description of one preferred embodiment embodying the present invention will now be given referring to the accompanying drawings.

A preferred embodiment of the present invention will be

described herein below by referring to Fig. 1 and Fig. 10.

[Network and hardware architecture for achieving the method for transforming data formats between database management systems in accordance with the present invention]

Now by referring to Fig. 1 and Fig. 2, a network and hardware architecture for achieving the method for transforming data formats between database management systems in accordance with the present invention will be described in greater details below.

Fig. 1 is a schematic block diagram of network architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention.

Fig. 2 is a schematic block diagram of hardware architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention.

As shown in Fig. 1, the network architecture of database management system in accordance with the present invention includes computers 100 (100A, 100B), a network 110, and a disk storage device 120. The computers 100 and the disk storage device 120 are connected by the network 110.

The computers 100 will issue a request of data

transformation to the disk storage device 120, which has a capacity enough to store entire data for a large-scale database. The disk storage device 120 suitable for use in the database management system in accordance with the present invention may have some degrees of intelligence to perform programs stored in the device in response to the instructions from the computers 100.

Components of this system will be described as follows, from the point of view of hardware architecture.

A computer 100A includes a processor 401A, a memory 402A, and host adapters 400A (400A-1, 400A-2). These components are connected via an internal bus 404A to send and receive commands and data.

The computer 100A is connected to the network 110 by the host adapters 400A.

The disk storage device 120 includes and host adapter controllers 410 (410A, 410B, 410C), a host-to-disk interface 420, disk adapters 430 (430A, 430B, 430C), and disks 200 (200A, 200B, 200C). The disk storage device 120 is connected to the network 110 by the host adapter controllers 410. The host adapter controllers 410 are connected to the disk adapters 430 inside the disk storage device 120. The disk adapters 430 is connected to a series of disks 200, which are so-called hard disk drives for storage a large amount of data using magnetic recording medium applied on an aluminium- or glass-based

substrate.

The host-to-disk interface 420 may access to any of arbitrary disks by configuring the network, or may configured so as to have some predefined disks to which each of host adapter controllers 410 can access.

A read/ write request from the computers 100 will be processed by the host adapter controllers 410, which will direct the read /write request to the disk adapters 430 which is connected to the desired disks 200 on which the requested data is stored.

[Software architecture of the method for transforming data formats between database management systems in accordance with the present invention and the operation thereof]

Now referring to Fig. 3 and Fig. 8, the software architecture and the operation of the method for transforming data formats between database management systems in accordance with the present invention.

Fig. 3 is a schematic block diagram of software architecture for achieving a method for transforming data formats between database management systems in accordance with the present invention.

Fig. 4 is a schematic diagram of transmission of request and reply to a program that achieves a method for transforming

data formats between database management systems in accordance with the present invention.

Fig. 5 is a flow chart illustrating the steps of a method for transforming data formats between database management systems in accordance with the present invention.

Fig. 6 is a schematic diagram of formats when a request 630 is achieved by packets of TCP/IP protocol.

Fig. 7 is a schematic diagram of a format when the request 630 is achieved by SCSI commands.

Fig. 8 is a schematic diagram of format of an I/O request 680.

As shown in Fig. 3, programs executed on the computers 100 are a skeleton program 600 and a communication program 610A. A host program table 620 is incorporated as a control table.

The skeleton program 600 is a program installed on a host in order to relay the data transformation requests from application programs to the disk storage device 120. The communication program 610A will perform interprocess communication between a communication program 610B on the disk device.

Programs executed on the disk storage device 120 are the communication program 610B, data transformer program 650, and I/O program 660.

The data transformer program 650 is a program for transforming data formats on the disk storage device 120, upon

reception of commands from the skeleton program. The method for transforming data formats between different database systems in accordance with the present invention features data transformation on the disk storage device 120, which in general is performed on the computers 100.

The I/O program 660 is a program controlling I/O to and from the actual disk devices, and the data transformer program performs data transformation by commanding the I/O program.

A storage program table 670 is incorporated in the I/O program 660 as a control table. The symbol FAL in the drawings designates to the "File Access Library", FCL to the "File Conversion Utility".

The data transformation steps of the database management system in accordance with the present invention will be described in greater details by referring to Fig. 3 and Fig. 4, following the steps shown in the flow chart of Fig. 5.

A request of data transformation from a user application program is issued to the skeleton program 600.

The skeleton program 600 upon reception of data transformation request will retrieve the address information on the data transformer program 650 from the host program table 620. The skeleton program 600 will then use the communication program 610A to transmit the transformation request 630A to the data transformer program 650 (step 1000). As an example of address of data format transformer program, there is a

combination of LUN (logical unit number) and port ID (port number) of a disk storage device, as shown in Fig. 3.

The communication program 610B on the disk storage will use the interprocess communication to send a transformation request 630B to the data transformer program 650.

The data transformer program 650, upon reception of the transformation request 630B (step 1010), will use the I/O program 660 to issue an I/O request 680 for reading out the source data to retrieve data as a reply 690 (step 1011).

The data transformer program 650 will then convert data to a predetermined format (step 1012) and write thus converted data to an address specified by the transformation request 630B, by issuing the I/O request 680 (step 1013). The data transformer program 650 will iteratively repeat this operation until no further data is present (step 1014). When there is no further data, the program will reply to the skeleton program the result of conversion process (step 1015).

Next, in the reverse order of request process, the data transformer program 650 will transmit a reply 640B to the communication program, which performs interprocess communication to deliver the reply 640B from the communication program 610A to the skeleton program 600 (step 1001).

At the time of interprocess communication, the address of the skeleton program 600 may be obtained from the storage program table 670.

The overview of the data structure of these requests and replies will be as follows.

When the computers 100 communicates with the disk storage device 120 to send and receive a request on TCP/IP packets, the request will be in the format shown in Fig. 6. The request format includes a medium-specific header 701, an IP header 702, a TCP header 703, a function ID 704, a request ID 705, and a parameter field 706.

Items arbitrarily set by the requesting user are the function ID 704, the request ID 705, and the parameter field 706.

The medium-specific header 701 contains information on the Ethernet, protocol of lower layer, and the like.

The IP header 702 contains information on IP protocol such as IP address. The TCP header 703 contains information on the port number.

The function ID 704 is an item for determining the function of request, and contains an identifier corresponding to "conversion" in case of data format conversion request.

The request ID 705 contains an identifier for determining uniquely a request.

The parameter field 706 of the request function is a field for storing parameters for this request. For example, as shown in Fig. 6, the source data address 706A, data size 706B, transformed data address 706C may be specified.

When the computers 100 and the disk storage device 120 send and receive a request using the Write command of the SCSI interface, the transmission will be in the format shown in Fig. 7.

When compared with the packet shown in Fig. 6, the difference is in the header. The header here is a SCSI Write command CDB (Command Description Block) 801. The items following are the same as those of Fig. 6.

The format of I/O request 680 to the I/O program 660 will be as shown in Fig. 8. An I/O request 680 contains a volume ID 901, an offset 902, a data size 903, and a memory address 904.

The volume ID 901 contains logical volume number of the disk. The offset is the offset of reading address of the device storing the data to be read out or the offset of writing address of the device to write data. The data size 903 is the size of data to be read out or written. The memory address is the destination memory address when transferring data read out of the device to the memory, or the offset address of the memory storing the data to be written into the device when writing to the storage device.

[An embodiment of the method for transforming data formats between different database systems in accordance with the present invention]

Now referring to Fig. 9 and Fig. 10, the method for transforming data formats between different databases systems will be further described by means of an embodiment by way of example.

Fig. 9 is a schematic block diagram of hardware and software architecture for the data transformation corresponding to Fig. 11.

Fig. 10 is a schematic block diagram of hardware and software architecture for the data transformation corresponding to Fig. 12.

The embodiment described with reference to Fig. 11 is a case of converting the data of DB format 1 on the disk 200A into the data of DB format 2 on the disk 200B, as have been described above.

The computers 100 are assumed to be Unix machines. On these Unix hosts the skeleton program 600 and the communication program 610A will be installed.

The host adapter controllers 410 of the disk storage device 120 will contain the communication program 610B, the data transformer program 650, and the I/O program 660.

The skeleton program will issue a request to the data transformer program 650 of the disk storage device 120 in response to the request from the application programs. The data conversion from the DB format 1 data to the DB format 2 data

will be performed by the data transformer program 650.

The embodiment illustrated in Fig. 12 is a case using two intermediate data formats of format 1 data and format 2 data, in order to convert the DB format 1 data on the disk 200A to the DB format 2 data on the disk 200B.

In this example, the computers 100 are assumed to be comprised of a mainframe computer 100A and a Unix host computer 100B.

The mainframe computer 100A will contain an extraction skeleton program 600A, and a communication program 610A. The Unix host computer 100B on the other hand will contain a transformation skeleton program 600B, and a communication program 610B.

Each of the disk adapter controllers 410A, 410B, 410C of the disk storage device 120 will contain an extraction program 650A, a transformation program 650B, and a load program 650C, respectively. The controllers will contain in addition a communication program and an I/O program.

When transforming the DB format 1 data of the disk 200A to the format 1 data of the disk 200B, the extraction skeleton program 600A receiving the request from an application program on the mainframe computer 100A, will issue a request to the extraction program 650A on the host adapter controller 410A.

In a similar manner, when transforming the format 1 data of the disk 200B to the format 2 data of the disk 200C, the

transformation skeleton program 600B, which receives the request from an application program on the Unix host 100B will issue a request to the transformation program 650B on the host adapter controller 410B. Also, when transforming the format 2 data of the disk 200C to the DB format 2 data of the disk 200D, the loader skeleton program 600C, which receives the request from an application program on the Unix host 100B, will issue a request to the transformation program 650C on the disk adapter controller 410C.

In this case in particular, data transformation of the Prior Art needs three round trips of data transfer between the hosts and storage, while in accordance with the present invention, no data transfer between the hosts and disk storage is required. The effect of the present invention is estimated to be significant.

[Effect of the present invention]

As have been described above, in accordance with the present invention, when transforming data formats between database management systems, a data transforming method of database management system may be provided, which may transform data within the disk storage device when transforming database data formats so as to reduce the system load.

Also, a data transforming method of database management

system, which may be easily developed by the program developer of the database management system and the program developer of the disk storage device may be provided.

5